A genetic procedure used to train RBF neural networks

Constantin-Iulian VIZITIU

Communications and Electronic Systems Department
Military Technical Academy
George Cosbuc Avenue 81-83, 5th District, Bucharest
ROMANIA
vic@mta.ro http://www.mta.ro

Abstract: - The performance level assigned to RBF neural networks used inside of pattern recognition systems depends a lot by their training algorithms and especially, by the specific techniques (e.g., clustering techniques) used for RBF center positioning. Starting from basic property of genetic algorithms to represent global searching methods, a full-genetic procedure to assure optimization both connectivity and neural weights of RBF networks is described. To confirm the broached theoretical aspects and having as starting point a real pattern recognition task, a comparative study (as performance level) with others standard RBF training methods and SART neural network is also presented.

Key-Words: - RBF neural networks, genetic algorithms, GA-based clustering techniques

1 Introduction

It is well-known that, RBF neural networks have their origin in the solution of the multivariate interpolation problem [1]. The RBF networks in their standard form have only one hidden layer (see Fig.1). Properly trained, they can approximate an arbitrary function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ mapping:

$$ f(x) = h(x) = w_0 + \sum_{i=1}^{m} w_i z_i(x), \quad (1) $$

where $x \in \mathbb{R}^n$, $\{w_i\}_{i=1,m}$ denotes the neural weights, $w_0$ is the bias and $z_i(x)$ represents the activation function (also known as radial basis function), which is given by equation:

$$ z_i(x) = \Phi \left( \frac{\|x - t_i\|}{\sigma_i} \right), \quad (2) $$

where $\|\|$ is the Euclidian norm, $t_i \in \mathbb{R}^n (i=1,m)$ is the center of RBF function, $\sigma_i$ is its width and finally, $\Phi(\cdot)$ is a nonlinear function that monotonically decreases (or increases) as $x$ moves away from $t_i$ (usually, by gaussian, cubic, multiquadratic type etc.). Note also that, a RBF neural network with more than one output can be used in specific applications [2].

Generally speaking, the training of RBF neural networks consists in two important stages, namely:

1) settlement of RBF parameters $\{t_i, \sigma_i\}_{i=1,m}$ (also known as RBF center selection or mapping);
2) using a proper (e.g., supervised etc.) training procedure, fitting of the neural weights to the output neural layer.

According to literature [1], [2], the most important step in this training procedure is represented by the selection of RBF centers, and several basic mapping strategies have been proposed in this field: random positioning, supervised selection (e.g., Cholesky, LU or QR decompositions, SVD technique etc.), and clustering techniques (e.g., k-means algorithm, fuzzy-c-means, SOMs, OLS algorithm, RAN algorithm, UCC algorithm etc.), respectively.

As is expecting, a more efficient approach in RBF center mapping employs a clustering technique, such
as for example, $k$-means, SOMs or other hybrid clustering algorithms [3], [4]. Let \( \{ (x_i, d_i) \}_{i=1}^{\overline{K}} \) be the set of the training examples, where \( x_i \) is an input vector and \( d_i \) its desired output. If at the ending of the applied clustering algorithm, the input vectors \( \{x_i\}_{i=1}^{\overline{P}} \) were assigned to \( K \) clusters \( S_1, S_2, \ldots, S_K \), then the mean vectors of each cluster given by equation:

\[
m_i = \frac{1}{|S_i|} \sum_{x_i \in S_i} x_i,
\]

become RBF centers (i.e., \( t_i = m_i, i = 1, \overline{K} \)). Finally, in order to calculate the widths \( \{ \sigma_i \}_{i=1}^{\overline{K}} \), several inexpensive heuristic approaches are available. For example, in [4], it is suggested that a single value \( \sigma \) for all basis functions gives good results (in fact, it was used \( \sigma = \langle t_i, t_j \rangle \), where \( t_i \) is the nearest center from \( t_j \) and \( \langle \rangle \) indicates the average over all such pairs). Other methods use a different value \( \sigma_i \) for each basis function. In [5], each width was defined according to equation:

\[
\sigma_i = \alpha \| t_i - t_j \|,
\]

where \( \alpha \) denotes an overlap factors.

Generally speaking, although the standard clustering algorithms ($k$-means, fuzzy-$c$-means, EM etc.) had been applied to many practical clustering problems successfully, it has been shown that these algorithms may fail to converge to a global minimum under certain conditions that represents certainly, an important disadvantage. Since genetic algorithms (GA) are powerful global searching tools, and are most appropriate for complex nonlinear models where location of the global solution is a difficult task, these algorithms can be applied with very good results, to evolve the proper number of clusters and to provide appropriate clustering, [3], [6].

Accordingly, this new important class of hybrid clustering methods synthetically called GA-based clustering algorithms, could offer an excellent opportunity to be implemented inside of RBF neural network training algorithm (e.g., as RBF center mapping).

This mixed approach combines the robust nature of the genetic algorithms with the high-performance of the standard clustering methods. As a results, in the literature, a lot of GA-based clustering techniques and their different applications are described (e.g., GKA algorithm (and its faster version, FGKA algorithm), IGKA algorithm, GGA algorithm, GKMODE algorithm, GCUK algorithm, GCDC algorithm, GWKMA algorithm etc. [7], [8], [9], [10], [11], [12]).

This paper is aimed to present a full-genetic training technique of RBF neural networks having as starting point a suitable GA-based clustering method used as center positioning, and a genetic approach used in fitting of the output neural weights, respectively. All these proposed objectives will be checked against a real pattern recognition task. Therefore, in the first part of the paper, a theoretical description of genetic procedure used in training of RBF neural networks is described. Next, a comparative study of the proposed genetic method with others standard RBF training techniques and SART neural classifier is indicated. Finally, in the last part of the paper, the most important conclusions about the approached aspects are also included.

2 Proposed GA technique

According to literature [3], [4], it is known that RBF networks represent as performance level, an efficient alternative to standard feedforward topologies (e.g., MLP neural networks), and the basic stages in their training process are represented by selection of RBF centers and fitting of the neural weights to the output layer, respectively. Generally speaking, the standard approaches used for RBF center positioning lead to some important drawbacks (e.g., different clustering methods can and do generate different solutions for the same dataset, improper behavior in case of very large datasets, problem of local minimums etc., [13]), but an optimal solution to increase the pattern classification performances of RBF networks could be represented by genetic selection of RBF centers [3], [14]. Finally, to obtain a full-genetic training procedure of RBF networks, the standard supervised or unsupervised methods used ordinarily to calculate the neural weights from the hidden layer to the output layer, can also be replaced with a proper genetic optimization technique.

Accordingly, to obtain both center positioning and training rule of a RBF neural network, the proposed optimization procedure contained the following two processing modules:

1) the task of the first processing module was to achieve the setting parameters of RBF \( \{ t_i, \sigma_i \}_{i=1,m} \), where \( m \) represents the number of centers (or hidden neurons). Accordingly, a GA-based clustering method containing the following basic steps is proposed:
s1) if the input training dataset is by the form 
\[ \{x_i, d_i\}_{i=1}^{P}, \quad x_i \in \mathbb{R}^n \] and \( K \) is the number (in this case, known) of the classes from the input space, then using an powerful (standard) clustering algorithm with zoom effect (e.g., \( k \)-means, fuzzy-\( c \)-means, ISODATA etc.), the most natural tendencies inside of each main data cluster were determined (i.e., each main data cluster was bounded into \( m_i \) new (sub)clusters where, \( m = \sum_{i=1}^{K} m_i \)). On the other hand, the basic rule to select the suitable clustering method was directly connected to the complexity (number of the clusters, cluster shapes etc.) of the input dataset. Finally, it can be observed that, using this first preliminary grouping procedure, a pre-clustering of the input space very useful for the next genetic optimization method, was thus obtained (see Fig.2); s2) the starting chromosome population was made using a random selection of \( m_i \) vectors \( x_i \) from each class (i.e., one vector for each bounded cluster) and finally, a suitable linear concatenation. Therefore, each chromosome had assigned \( m \) vectors \( \{\eta_i\}_{i=1}^{m} \) which are extracted from the input dataset. To achieve a proper chromosomal representation, a real encoding technique was also used.

![Fig.2: Genetic encoding technique used in case of RBF center positioning](image)

The fitness used for each chromosome evaluation was calculated according to standard equation:

\[
E = \left[ 1 + \left( \frac{1}{P} \sum_{i=1}^{P} (y_i - d_i)^2 \right)^{0.5} \right]^{-1},
\]

where \( y_i \) denotes the RBF network real output.

The stopping criterion was represented by the exceeding of the maxim generation number (this number has a constant value) or when the goal error was reached. The selection of the parents for the next chromosomal generation was also made using the well-known roulette principle.

The continuous crossover supposed the use of two splitting points (randomly chosen), and each chromosome had attached a certain crossover probability with values into \([0.5, 0.85]\) range. To introduce new individuals inside of the current population and to protect GA against irreversible and accidental information failures generated by improper crossover operations, the uniform mutation operator was also used.

To certainly obtain the best chromosomal solution offered by the proposed GA, a method similar with Gallant (or pocket) algorithm from neural network theory was used [2].

After the applying of RBF center selection procedure, \( \{\sigma_i\}_{i=1}^{m} \) width for each hidden neuron was calculated according to standard equation:

\[
\sigma_i^2 = \frac{1}{P} \sum_{x_i \in S_i} (x_i - t_i)^T (x_i - t_i), i = 1, m,
\]

where \( S_i \) is the bounded cluster assigned to center \( t_i \) and \( P_i \left( \sum_{i=1}^{m} P_i = P \right) \) is the number of training vectors \( x_i \) from this data subcluster.

Because in this moment RBF setting parameters \( \{t_i, \sigma_i\}_{i=1}^{m} \) are known, the neural weights to the output layer \( \{w_{ij}\}_{i=1}^{K}, j=1, m \) will be calculated using the second processing module which will be below described.

More details regarding the structure of GA used for RBF network center mapping can be found in [3]. m2) usually, a RBF neural network is trained using either standard supervised methods based in fact, on gradient descendent algorithm or unsupervised techniques (e.g., OLS algorithm etc.). Even though their popularity is indubitable, these training algorithms have some major drawbacks, for example in case of supervised procedures, one of the most important being related to its local optimization character. Although some improved versions of these algorithms are indicated in literature [3], [4], an efficient solution to remove their basic disadvantages can be represented by the genetic algorithm use (e.g., it is known that, genetic algorithms represent...
powerful searching tools into complex data space, and which offer solutions by global type).

Having as starting point the RBF network parameters which was before determined (i.e., the best chromosomal solution offered by the first genetic module), the task of the second module was to optimize the distribution of the neural weights to the neural output layer. Consequently, for a suitable chromosomal representation of the neural weights, these were random initialized, and were real encoding into a linear structure so that, each chromosome represents a single output weights set (see Fig.3). Because the topology is predefined and remains fixedly after initialization of the training process, a single chromosome will contain only the values of neural weights and do not incorporate any information about its connectivity.

![Genetic encoding technique used in case of neural weights fitting](image)

**Fig.3: Genetic encoding technique used in case of neural weights fitting**

The fitness used for each chromosome evaluation was calculated according to standard equation:

$$E = \left( \frac{k}{1 + \text{MSE}} \right) \left( \frac{\text{max number of cycles}}{\text{current cycle}} \right) = ct, \quad (7)$$

where MSE error was estimated for all \( \{x_i\}_{i=1}^{P} \) training patterns.

The stopping criterion was similar with the one used in case of the first genetic module. The selection of the parents for the next chromosomal population was also made using the well-known roulette algorithm.

The continuous crossover supposed the use of two random splitting points, and each chromosome had attached a certain crossover probability with values into [0.6, 0.95] range. By the same reasons as in case of first genetic module, the uniform mutation operator was also used.

To certainly obtain the best chromosomal solution, a procedure quite similar with the pocket algorithm was also implemented.

More details related to the structure of the second genetic module used for optimization of the neural weights to output neural layer can be found in [3].

### 3 Experimental results

To add more consistency to the theoretical aspects treated in the first part of this paper, a real pattern recognition (PR) task was proposed to be solved (i.e., the PR task was to classify using thermal imagery, 5 input classes representing FLIR images of some well-known military aircrafts (see Fig.5)). Consequently, the basic diagram (setup) used to generate the real thermal database for testing of the proposed genetic technique, is presented in Fig.4.

![Database design](image)

**Fig.4: Database design**

The acquisition and preprocessing algorithms used in case of thermal imagery were similar with ones described in [2]. Accordingly, a number of 40 thermal images/class was obtained. To implement the feature selection stage, the modified Flusser invariants (by moment’s type) proposed in [3] were achieved. Finally, the feature vectors matrix for the next classification purposes had 11×200 dimension.

The main objectives of this experimental paragraph were:

- o1) to demonstrate the superiority at the performance level, of the proposed genetic optimization procedure comparing to others standard approaches used to train a RBF neural network;
- o2) to compare at the performance level, the classification chain incorporating proposed optimized RBF network with the chain made by use of SART neural network.

To quantify and compare the performance level of different tested PR chains, as most important indicator, the classification rate (CR) has been computed. It is known that, the CR represents, in [%], the ratio between the number of correct classified patterns and the total number of these.
Related to the first experimental objective, the full-genetic procedure used to optimize both connectivity and neural weights of RBF networks was compared as performance level (i.e., CR, training time etc.), to other standard (i.e., a supervised procedure for RBF center mapping and OLS algorithm, respectively) and improved (i.e., UCC algorithm proposed by Brown, [3]) training techniques.

Using the available thermal image database, the CRs and other important parameters obtained after comparative study of these training methods, are synthetically presented in Table 1. Supplementary, in case of the proposed genetic technique, a minimal 2D projection of RBF network center mapping over input data space is shown in Fig.5.

Using the previous testing conditions, the CRs and other important parameters obtained after comparative study between the two classification chains made on one hand, by optimized RBF network and, on the other hand, by SART network are synthetically indicated in Table 2. In addition to these experimental results, a graphical illustration of the input vectors classification made by two neural networks is shown in Fig.6.

**Table 1**

<table>
<thead>
<tr>
<th>Center selection</th>
<th>Classification performances (CR and others training parameters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>supervised mapping</td>
<td>90%</td>
</tr>
<tr>
<td>m=8; K=5; 0.1 s; nepochs=10⁵; ε=10⁻⁴; σ=1</td>
<td></td>
</tr>
<tr>
<td>OLS algorithm</td>
<td>92%</td>
</tr>
<tr>
<td>m=10; K=5; 0.15 s; nepochs=10⁵; ε=10⁻⁴; σ=1</td>
<td></td>
</tr>
<tr>
<td>UCC algorithm</td>
<td>93.5%</td>
</tr>
<tr>
<td>m=11; K=5; 0.21 s; nepochs=10⁵; ε=10⁻⁴; σ=0.8</td>
<td></td>
</tr>
<tr>
<td>proposed GA</td>
<td>95.5%</td>
</tr>
<tr>
<td>m=14; K=5; 0.67 s</td>
<td></td>
</tr>
<tr>
<td>nepochs=10⁵; ε=10⁻⁴; σ=0.8</td>
<td></td>
</tr>
<tr>
<td>GA modules</td>
<td></td>
</tr>
<tr>
<td>155 s maxpop=50; maxgen=100</td>
<td></td>
</tr>
<tr>
<td>121 s maxpop=50; maxgen=75</td>
<td></td>
</tr>
<tr>
<td>p₀=0.8; ε₀=10⁻²</td>
<td></td>
</tr>
<tr>
<td>k=0.75</td>
<td></td>
</tr>
</tbody>
</table>

Having as starting point the experimental results reported in Table 1, a first preliminary conclusion is that one, the proposed genetic optimization procedure leads to an increased (on average) CR generally 3.6% more than other RBF network centers selection approaches. As can be seen in Fig.5, the first genetic module also leads to a very good center mapping over input data space (i.e., each significant data cluster had allocated at least a RBF center) even through it was used only a minimal 2D projection.

According to [16], an important property of SART network is that it needs no initial system parameter specifications and no prespecified number of center vectors. Because the number and the final values of the prototypes are automatically found during the training process for the SART network, this neural classifier represents in fact, an improved version of RBF (or LVQ) networks trained by standard approaches. More by token, the second experimental objective is one as soon as justified.

**Experimental results**

Having as starting point the experimental results reported in Table 2, a second preliminary conclusion is that one, the CRs obtaining in case of two tested
neural networks are on average, similar. However, optimized RBF network turns out to be faster than SART network as training time and thus, more suitable to be (FPGA) hardware implemented [17], inside of real classification chain (e.g., ATR, ATTR systems etc.).

The applications described in this paragraph were implemented using specific functions from MATLAB toolboxes nnet, image processing and GAOT on a Pentium™ processor at 2.4 GHz.

More details regarding experimental aspects treated in this section can be found in [2] and [17].

4 Conclusions

The theoretical and experimental results presented in this paper lead to the following important remarks:

c1) the full-genetic procedure proposed to train RBF neural networks turns on to be as performance level, a correct option comparing to others standard (clustering) training techniques. In another train of thoughts, using a genetic searching of the RBF network parameters (i.e., connectivity, weights), the optimal property of this method is also assured;

c2) the optimized RBF neural network provides a classification level comparable with one assured by SART neural network, but this classifies much faster than the last one;

c3) finally, inside of (real-time) PR systems, the genetic optimized RBF network assures very good classification results (i.e., CRs more than 95%) and fully justifies as future potential option, its next hardware implementation.

References:


